TITLE OF THE INVENTION

A METHOD OF EDITING STURCTURED DOCUMENTS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to document editing techniques. More specifically, the present invention relates to document editing techniques for extracting elements of structured documents and generates a desired output document.

2. Description of the Related Art

It is known in the art to extract necessary elements from a plurality of documents and deal them for generating an output document. A conventional editing technique for obtaining such an output document is disclosed in Japanese Laid-open Paten Application No. 6-259421. According to this conventional technique, the elements of the structured input document are extracted using matches of character sequences with the elements in the document, a sequence connector, a hierarchy connector, etc.

However, the aforesaid conventional technique is not provided with an element connector and thus, it is undesirably required to use one character pattern for establishing a match with the element in the document and extracting the matched element. In other words, if a plurality of different elements are to be extracted, the same number of different character patterns are necessary.

Further, the aforesaid conventional technique is unable to extract a plurality of elements while maintaining relationship of the elements extracted. This is because the elements extracted using different character patterns are independent with one another.

The conventional technique in question is provided with a sequence connector for extracting a plurality of elements in sequence. However, the conventional technique is not provided with any connector via which a plurality of elements are extracted in parallel in any order. Therefore, it is difficult to extract a plurality of elements from input documents if the arrangement order of elements is

different with each document.

SUMMARY OF THE INVENTION

It is therefore an object of the present to provide to overcome the above-mentioned difficulties of the conventional editing techniques.

SUB B2 In brief, these objects are achieved by a technique of editing a plurality of structured documents is disclosed. A plurality of structured documents are inputted in a document edit system. Thereafter, a plurality of elements are extracted from each of the plurality of structured documents using an element edit statement which indicates element to be extracted. The extraction of the elements is implemented while the relationship of the elements extracted is maintained.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become more clearly appreciated from the following description taken in conjunction with the accompanying drawings in which like elements are denoted by like reference numerals and in which:

Figs. 1 to 9 are each diagram for describing connectors and notations used in the present invention;

Fig. 10 is a diagram showing a document edit system used in a first preferred embodiment of the present invention;

Fig. 11 is a diagram showing input and output documents together with the functional block of the system of Fig. 10;

Fig. 12 is a diagram showing two structured documents that respectively correspond to the input documents of Fig. 11;

Fig. 13 is a diagram showing a structured document that corresponds to the output document of Fig. 11;

Figs. 14 to 17 are each flow chart which includes the steps which characterize the operation of one example of the first embodiment;

Fig. 18 is a diagram showing a structured document used in another example of the first embodiment;

Fig. 19 shows a diagram showing a document edit system used in a second preferred embodiment of the present invention;

Fig. 20 is a diagram showing input and output documents together with the functional block of the system of Fig. 19;

Fig. 21 is a diagram showing a structured documents that corresponds to the output document of Fig. 11;

Figs. 22 to 28 are each flow chart which includes the steps which characterize the operation of the second embodiment;

Fig. 29 shows a diagram showing a document edit system used in a third preferred embodiment of the present invention;

Fig. 30 is a flow chart which includes the steps which characterize the operation of the third embodiment; and

Figs. 31(A) to 31(C) are each showing an output document obtained in the third embodiment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Preferred embodiments of the present invention will be described with reference to the accompanying drawings. Throughout the instant disclosure, it is assumed, for a convenience of description, that the document to be processed or edited is an SGML structured document. However, the present invention is applicable to the documents other than SGML ones on the condition that the document is tagged or hierarchically structured.

Before turning to the concrete embodiments, it is preferable to define edit operators and notations, both used in the instant invention.

(a) Tag:

Tag is delimited using the '<' and '>' characters. Tags are used to define an element which is an identified component of a document. The element usually consists of a start-tag, content and an end-tag. However, an element may involve one or more elements. The end-tag is delimited by '</' and '>'. Element matching operation using the tag is briefly described with reference to Fig. 1. As shown in Fig. 1, an element edit statement 10 includes a tag <note>. A reference numeral 12 denotes a structured document. The tag <note> in the element edit statement

10 is used to implement the element matching with one or more elements involved in a document portion 14 that is an document area to be searched. If the document portion 14 contains an element named <note>, a match is established.

(b) Character Pattern:

The character pattern, which consists of normal text characters (such as typically used in UNIX commands) of a character sequence, is delimited or defined using the " (left double quotation mark right) and " (right double quotation mark). The character pattern is used to search for document data which includes at least one character sequence defined by the character pattern. That is, the existence of document data is ascertained if a match is established between the character pattern and the document data. In the instant disclosure, the document data signifies a document portion excluding tags. One example of matching using character pattern is described with reference to Fig. 2. As shown in Fig. 2, the element edit statement 10 includes a character pattern "the". The structured document 12 involves an element defined by <ooo> and </ooo> which contains a character sequence of "the" and thus, a match is established between character pattern "the" and the document data within the element defined by <ooo> and </ooo>.

(c) Wild Card Tag <*>:

The wild card tag is defined by the '*' character delimited using '<' and '>', and matches each of document elements hierarchically structured. One example of matching of the wild card tag <*> with the document elements is shown in Fig. 3. As shown in Fig. 3, the element edit statement 10 includes the wild card tag <*>. The structured document 12 includes a document portion 16 to be searched for editing, which portion includes a hierarchy formed by "chapter", "section", and "paragraph". In the case shown in Fig. 3, the following matches are established.

(c-1) The wild card tag <*> matches "zero" hierarchical layer (viz., matches "vacant" structure) in connection with "chapter".

(c-2) The wild card tag <*> matches the hierarchical layer (viz., element) of "section".

(c-3) The wild card tag <*> matches the hierarchical layer (viz., element) named "paragraph" within the element of "section".

(d) Negation Indicator '!':

The negation indicator is defined by the '!' character (exclamation character). The negation indicator is used to specify an element wherein a match is not established with a character sequence immediately following the negation indicator. Fig. 4 is a diagram schematically showing one example of matching of a character sequence "this" preceded by the negation indictor with an element in a document portion 18 of the structured document 12. The character sequence "the" is defined in the element edit statement 10.

(e) Extraction Indicator '%':

The extraction indicator is defined using the '%' character, and is used to extract an element from the structured document, which element matches an element designated by a tag which follows the extraction indicator '%'. One example of the usage of the extraction indicator '%' is described with reference to Fig. 5. As shown in Fig. 5, the element edit statement 10 contains a tag <note> which is preceded by the extraction indicator '%'. The structured document 12 involves the document potion 14 that is currently to be searched. In this case, the element including tags <note> and </note> is specified and then extracted.

(f) Sequence Connector ',':

The sequence connector is defined by the ',' (comma) character. Condiering the example of <A>,<B>, in the case of which the element specified by tag A must precede the element specified by tag B. However, a match is also established even if another element exists between the elements respectively defined tags A and B. One example of the sequence connector ',' is shown in Fig. 6. As shown, the element edit statement 10 contains the sequence connector ',' between tags <beginning> and <ending>. The structured document 12 involves a document portion 20 which contains the elements "beginning" and "ending" in this order. As a result, matches are established as illustrated.

(g) Hierarchy Connector:

The hierarchy connector is defined by inserting no character between adjacent tags. A match is established only if the element (defined by the first tag and denoted by C) preceding the hierarchy connector contains the element (defined by the second tag and denoted by D) following the hierarchy connector in the document portion to be searched. It is to be noted that a match is also established even if there exists another element between the elements C and D. One example of usage of the hierarchy connector is described with reference to Fig. 7. As shown in Fig. 7, the element edit statement 10 contains the hierarchy connector between two tags <section> and <paragraph>. The structured document 12 involves a document portion 16 which is currently to be searched for and contains the hierarchical data structures <chapter>, <section> and <paragraph>. In this case, the element <section> contains the element <paragraph> and thus, a successful match is established as illustrated.

(h) Parentheses:

The parentheses '(' and ')' indicate that the element(s) within the parentheses is preferentially processed.

(i) AND Connector '&':

The AND connector is defined using the '&' character. The example '(E & F) indicates that 'F' may either follow or precede 'E' in the document portion to be searched for editing. Referring to Fig. 8, one example of usage of the AND connector is shown. As shown, the element edit statement 10 contains the AND connector '&' sandwiched by two tags <beginning> and <ending>. The structured document 12 contains the elements, defined by the tags <beginning> and <ending>, in the document portion 20 and thus, a successful match is established as illustrated.

(j) OR Connector '|' :

The OR connector is defined by the '|' character. The example (G | H) indicates that 'G' may be present or 'H' may be present in the document portion to be searched for editing. The OR connector (|) is further described with reference to Fig. 9. As shown, the element edit statement 10 contains the OR connector

sandwiched by two tags <beginning> and <ending>. The structured document 12 contains only the element <beginning> in the document portion 22 and thus, one successful match is established as illustrated.

Referring to Fig. 10, there is shown a structured document edit system 30 of a first embodiment of the present invention in block diagram form.

As shown in Fig. 10, a controller 32, which typically takes the form of central processing unit (CPU), is provided to control the overall operation of the system 30. A memory 34 is arranged so as to receive a plurality of documents, on a one-by-one basis (usually), from a data base 36 which has stored a plurality of structured documents. A memory 38 is provided for storing one or more element edit statements. Typically, one statement is applied to the memory 38 via a suitable input interface 40.

A document edit engine 42 accesses the memories 34 and 38, and edits the document stored in the memory 34 using the element edit statement, and applies the edit results (in sequence or in batch) to a memory 44 to store the same therein. The edit result is adaptively outputted using a document retriever 46 and an appropriate output interface 48. In this case, the edit result may be arranged using an edit result arranging data applied to the edited document retriever 46.

A first example of the first embodiment will be described with reference to Figs. 11-17.

Fig. 11 is a diagram which shows a plurality of input documents (only two documents 50 and 52 are shown), an output document 54, and the document edit system 30 (Fig. 10). The output document 54 is a list showing the edit result after being subject to listing order arrangement. The first example of the first embodiment is to edit a plurality of input documents, including the input documents 50 and 52, so as to, in this particular case, extract the names of paper's authors and corresponding paper's titles, after which the edit (extracting) results are listed.

As shown, the input document 50 contains the following items.

Summary of paper

Name of Society: ABC Meeting

Title: DEF Report
Name: Taro SATO
Name: Hanako SUZUKI
Abstract: Until recently, ...

sub B8 On the other hand, the input document 52 contains the following items.

Summary of paper
Name of Society: GHI Meeting
Title: Analysis of JKL
Name: Jiro NAKAMURAO
Abstract: This report, ...

The output document 54 contains the following items which are results of document editing carried out by the edit system 30.

List A

# Name	Title
Taro SATO	DEF report
Hanako SUZUKI	DEF report
Jiro NAKAMURA	Analysis of JKL
...	...

sub B9 Fig. 12 shows the structured input documents 50' and 52' that respectively correspond to the input documents 50 and 52, while Fig. 13 shows the structured document 54' that corresponding to the output document 54. It is understood that the content of each of the items ("SUMMARY OF PAPER", "NAME OF SOCIETY", etc.) is defined using a start-tag defined by '<' and '>' and an end-tag defined by '</' and '>'. The detailed descriptions of Figs. 12 and 13 are deemed redundant and accordingly, will be omitted for brevity.

The operation of editing the input documents 50 and 52 and generating the output document 54, will be described using Figs. 14-17 together with Fig. 10-13.

sub B10 Fig. 14 is a flow chart which shows the steps which characterize the overall operation of the document data editing according to the first embodiment. In Fig. 14, at step 60, the first input document 50 is applied to the memory 34 form the

database 36. Assuming that an element edit statement is:

%<title>.%<name> ... element edit statement A

which is referred to as the element edit statement A for the sake of simplifying the discussion. At step 62, the element edit statement A is written into the document edit engine 42, after which the routine proceeds to a sub-routine 64 for element extraction. The sub-routine 64 is shown in Fig. 15 in detail.

Referring to Fig. 15, at step 66, if the element edit statement, retrieved into the document edit engine 42, contains one or more than one "OR connectors", the element edit statement is divided into a plurality of element edit instructions (operators). such as %<title> and %<name>, every "OR connector". However, in the instant case, there exists no "OR connector" in the element edit statement A and thus, the step 66 is not executed. At step 68, one of the element edit instructions divided at step 66 is selected. However, as mentioned above, no division of the statement is carried out at step 66 and thus. the element edit statement A is selected as a whole, after which the routine goes to a sub-routine 70, the details of which is shown in Fig. 16.

Referring to Fig. 16, at step 72, the element edit statement containing "AND connector (&)" is processed. Although no "AND connector" is contained in the element edit statement A, the flow chart of Fig. 16 is described in that another example, set forth later, includes "AND connector". At step 72, if the element edit statement, applied to the document edit engine 42 (Fig.10), contains one or more "AND connectors", the statement is divided into a plurality of element edit instructions or tags every each "AND connector". However, in the instant case, no "AND connector" is contained in the element edit statement A and thus, the step 72 is not executed. At step 74. one of the element edit instructions or tags, divided at step 72 is selected. However, as mentioned above, no division of the statement is carried out at step 72 and thus, the element edit statement A is selected as a whole, after which the program goes to a sub-routine 76, the details of which is shown in Fig. 17.

In Fig. 17, at step 78, the first element edit instruction (or tag) of the element

edit statement A is selected for execution. Following this, the program proceeds to step 80 at which a check is made to determine if the edit instruction (or tag) selected at step 78 (viz., <title> in the instant case) is parenthesized. The element edit tag <title> is not parenthesized and thus, the routine goes to step 82 at which the tag <title> is processed to determine if the element "title" matches any element of the structured document 50'. In the instant case, a match is established with the following element in the structured document 50':

<title>
DEF Report
</title>

Subsequently, at step 84, a check is made to determine if the matching execution has completed. In this case, since the answer to the inquiry made at step 84 is negative, the routine proceeds to step 86. The element edit statement contains the sequence connector ',' and hence, at step 86, the document portion following the element which has matched at step 82, becomes the next document portion to be searched. Following this, at step 88, the next element edit instruction (tag) (viz., <name>) is selected and the routine goes back to step 80. Thus, the following elements in the structured document 50' are matched:

<name>
Taro SATO
</name>

and

<name>
Hanako SUZUKI
</name>

Since the matching execution in connection with the document 50' completed, the answer to the inquiry made at step 84 is positive. Thus, the routine goes to step 90 at which a check is made to determine if any match has been established. In this case, the matches are established as mentioned above, the routine proceeds to step 92 at which a check is further made to determine if the extraction indicator '%'

exists the statement A. At step 94, since the extraction indicator '%' is involved in the element edit statement A, the matched elements are extracted and stored in the memory 44. Thereafter, the routine goes to step 96 of Fig. 16. In the above, if the answer at step 80 is positive, the parentheses are deleted at step 81 and the routine jumps to the sub-routine 64 (Fig. 15).

After executing step 94 (Fig. 17), the routine goes to step 96 at which a check is made to determine if all the element edit instructions (or tags), divided at step 72, have been processed. In the instant example, the element edit statement A does not involve any AND connector and hence, steps 96, 98, and 100 are not executed and the routine goes to step 102 (Fig. 15). In the instant example, the element edit statement A does not involves any OR connector and hence steps 102, 104 and 106 (Fig. 15) are not executed.

Subsequently, the routine goes to 108 (Fig. 14) at which a check is made to determine if the element edit statement to be processed remains. Since no element edit statement to be processed with respect to the document 50' remains, the routine goes to step 110. The document 110 has not yet been processed and hence, the routine goes back to step 60 at which the next document 52' is inputted to memory 34, after which the above mentioned processing is carried out in the same manner. The processing of the document 52' is clear from the foregoing, the description thereof will be omitted for brevity. It is understood that the element extracted from the document 52' is:

<title>
ANALYSIS OF JKL
</title>

and

<name>
Jiro NAKAMURA
</name>

All the elements thus extracted are stored in the memory 44 as the edit result. Thereafter, the edit result is retrieved by the edited document retriever 46, which

arranges and add data (or name(s)) if the edit result arranging data is applied thereto. In the instant example, the name "LIST A" is added to the top and, the names of the authors are listed at the left side of the list. Subsequently, the document named "LIST A" is outputted from the system 30. The above operations are implemented at step 112.

A second example of the first embodiment will be described. This example extracts the items of "name of society" and "title" from the documents 50 and 52, and outputs the edited document as a list B which is shown below.

LIST B

NAME OF SOCIETY	TITLE
ABC MEETING	EDF REPORT
CHI MEETING	ANALYSIS OF JKL

In this example, the element edit statement is:

%<society>&%<title> ... element edit statement B

As shown, the above statement is referred to as "element edit statement B" for the sake of convenience of description. This example features that the elements redefined by the tags in the statement B are indenpently matched and then extracted. Consequently, even if the items in the input documents 50 and 52 are differently arranged, the element extraction can be implemented without difficulty.

In the second example, the tags <society> and <title> are divided at step 72 (Fig. 16) and then the matching of the elements. defined by the tags <society> and <title>, in each of the documents 50' and 52' are able to be carried out independently. The extracted items (elements) are arranged, using the edit result arranging data applied to the edited document retriever 46 (Fig. 10), so as to meet the requirement shown above. Further, the name "LIST B" can be added to the output document using the aforesaid edit result arranging data.

Sub B13 A third example of the first embodiment will be described, which includes OR connector. An input document, indicated as a structured document, is shown in Fig. 18 and denoted by numeral 120. As shown in Fig. 18, an outermost element is a paper element which contains a first-p (first paragraph) element and a second-p

(second paragraph). Further, the first-p element contains a figure element, and similarly, the second-p element contains a figure element. The third example is to extract the figure element from each of the first-p and second-p. Thus, the element edit statement in the third example is:

(<first-p> | <second-p>)%<figure> ...element edit statement C

As shown, this statement is referred to as "element edit statement C" for the sake of convenience of description. It is to be noted that (<first-p> | <second-p>) and %<figure> are connected by the hierarchy connector since no character is provided therebetween. The third example contains OR connector and thus, the tags within the parentheses are divided at step 66 (Fig. 15), after which these tags are separated and are subject to the element matching process on an element-by-element basis. As mentioned above, the hierarchy connection is used, the element <figure> of each of the elements <first-p> and <second-p> is extracted.

A second embodiment of the present invention will be described with reference to Figs. 19-28.

Referring to Fig. 19, there is shown a structured docment edit system denoted by 30' which, in addition to the arrangement shown in Fig. 10, comprises the following new blocks.

(1) block 130: document edit engine 2.

(2) block 132: memory for storing an edited document.

(3) block 134: document content changing means.

(4) block 136: sort engine.

(5) block 138: memory for storing one or more initializing instructions.

(6) block 140: memory for storing one or more pre-edit instructioins.

(7) block 142: memory for storing one or more post-edit instructions.

(8) block 144: memory for storing one or more editted document arranging instruction(s).

In order to differentiate the block 42 from the block 130, the block 42 is indicated by the document edit engine 1.

Fig. 20 shows two input documents 146 and 148, the edit system 30' of Fig.

19. and an output document 150. The input documents 146 and 148 are respectively identical with the documents 50 and 52 of Fig. 11, and thus. the further descriptions thereof will be deemed redundant and accordingly omitted for simplifying the instant disclosure. The ouput document 150 is a list C which includes the title of the paper and the number of authors, which are extracted from the documents 146 and 148. Further. the total number of the authors are calculated and listed. Still further, the name "title" in each of the input documents 146 and 148 is changed to "Title of Paper" in the list C, while the names "Number of Authors" and "Total Number" are added in the list C.

Fig. 21 shows a structured out document 150' which corresponds to the output document 150 of Fig. 20.

In order to generate the list C shown in Fig. 20, element edit statements D-1 and D-2 are inputted to the memory 38. More specifically,

The element edit statement D-1 contains:

(1) Edit statement 1 which is %<title> for extracting the element <title>;

(2) Edit statement 2 for changing the name "title" to "Titel of Paper" and storing the changed name in a variable "title".

The element edit statement D-2 contains:

(1) Edit statement 3 which is %<name> for counting the number of authors:

(2) Edit statement 4 which counts the number of extracted names and stores the counted number in a variable "count".

The operation of the second embodiment will be described with reference to Figs. 19-28. In Fig. 22, an operation starts with a sub-routine 200 the detail of which is shown in Fig. 23. At step 200 (viz., sub-routine 200), the initializing instructions, which are retrieved from the memory 136, includes two instructions, one of which is to reset a variable "total" which stores the total number of the authors and the other of which is to reset a variable "table" provided for arranging the edit result and generating the list C. At step 202, one input document is applied to the system 30', after which a pre-edit operation is implemented at step 204 (viz., sub-routine 204). Following this, a plurality of operatioins for generating the output

document 150' are carried out at the following steps as shown in Figs. 22-28.

A third embodiment of the present invention will be described with reference to Figs. 29, 30, and 31(A)-31(C). The structured document edit system 30" of Fig. 29 differs from that of Fig. 19 in that the former arrangement further comprises an input document delection request receiver 298. Fig. 30 is a flow chart which shows the steps which characterizes the operation of the third embodiment. On the other hand, each of Figs. 31(A)-31(C) shows an output document generated from the system 30".

It will be understood that the above disclosure is representative of only three possible embodiments of the present invention and that the concept on which the invention is based is not specifically limited thereto.